Universidad
Andrés Bello

PUNTEROS A ESTRUCTURAS

Ingeniería en Computación e Informática

Estructuras con punteros

- En este caso, la estructura requiere almacenar una cantidad de elementos que se saben en tiempo de ejecución.
- Para ellos usamos un puntero.
- Para el ejemplo, usamos una lista de notas y la cantidad de ellas.
- Para acceder a los elementos de dicha lista, se acceden de forma habitual, con "*", "(", ")" y "." (punto).

Observaciones:

- nta no es puntero por lo tanto se accede a las columnas de la estructura Nota con "." (ejemplo: nta.cantNotas y nta.notas

- se usa nta.notas encerrado entre *( ) porque la variable notas de la estructura Notas es una lista (o un vector) dinámico

```c
#include <stdio.h>
#include <stdlib.h>

//Estructura con 2 columnas: un puntero que guarda notas y
//                           la cantidad de Notas
typedef struct nota{
    int *notas; //puntero a un arreglo
    int cantNotas;
}Nota;

int main( int argc , char *argv []) {
   Nota nta; //nta es del tipo Nota (estructura)
   int i;
   //Asignar valor a cantNotas
   nta.cantNotas = 3; //nta. permite acceder a las columnas de la estructura

   //Se reserva memoria para arreglo dinámico notas según cantNotas
   nta.notas = malloc(nta.cantNotas * sizeof(int));

   //Asignar las tres notas al arreglo
   *(nta.notas) = 6; //es lo mismo que nta.notas[0] = 6;
   *(nta.notas+1) = 5; //es lo mismo que nta.notas[1] = 5;
   *(nta.notas+2) = 4; //es lo mismo que nta.notas[2] = 4;

   // Muestra las notas
   printf("Las notas son:\n");
   int n = nta.cantNotas;

  for (i=0; i<n ; i++) { //Recorre cada fila i
     printf("Nota i: %d\n", *(nta.notas+i) );
   }
   return 0;
}
```

Punteros a estructuras

- Se reserva en forma dinámica elementos del tipo de una estructura.
- Para poder acceder a ellos, se realiza de dos formas principalmente:
  - Operador flecha: `->`: Accede directamente a los elementos.
  - Operador "`.`" y `*`: Se utiliza el operador de indirección (`*`) para ingresar a la estructura y después con el . para poder acceder al elemento.
  - Operador "`.`" y `[]`: se maneja como si fuera un arreglo.

Puntero a estructura a estructura

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/*Estructura llamada Nota con 2 variables:
  - un puntero que guarda notas (arreglo dinámico)
  - y la cantidad de Notas que se van a guardar
*/
typedef struct nota{
    int *notas; //puntero a un arreglo de notas
    int cantNotas;
}Nota;

/*Otra Estructura llamada Alumno con 3 variables:
  - un string que almacena el nombre del alumno
  - la edad del alumno
  - y un puntero (llamado pnota) que apunta a
    la estructura Nota,que a través de este puntero
    se podran almacenar las notas del alumno
*/
typedef struct alumno{
    char nombre[50]; //arreglo largo fijo (no dinámico)
    int edad;
    Nota *pnota; //pnota apunta a la estructura Nota.
}Alumno;
```

```c
int main( int argc , char *argv []) {
   int i, j;
   Alumno *alum; //Arreglo dinámico de tipo Alumno
   int n=3; //sev puede también leer n por teclado
   alum = malloc(n * sizeof(Alumno)); //reserva memoria para los n alumnos

   for (i=0;i<n;i++) {
      printf("INGRESE DATOS DEL ALUMNO %d:\n ", i );
      printf("Nombre: ");
      scanf ("%s",(alum+i)->nombre); //Nombre del alumno i
      printf(" Edad  : ");
      scanf ("%d",&(alum+i)->edad); //edad del alumno i

      //INICIALIZACIÓN DEL PUNTERO pnota (que apunta a la estructura Nota)
      Nota *ntaAlum = malloc(sizeof(Nota)); //reserva memoria para la estructura que almacena las notas
      (alum+i)->pnota = ntaAlum; //El puntero pnota debe apuntar a la derección reservada anteriormente

      printf("Cantidad de Notas: ");
      scanf ("%d", &(alum+i)->pnota->cantNotas);

      (alum+i)->pnota->notas =  malloc( (alum+i)->pnota->cantNotas * sizeof(int));

       for (j=0; j < (alum+i)->pnota->cantNotas ; j++) {
          printf("Nota %d: ", j);
          scanf ( "%d", (alum+i)->pnota->notas+j );
       }

   }//TERMINA INGRESAR ALUMNOS

   // Muestra las notas de cada alumno
   printf("\n ********************* ");
   printf("\n    SALIDA DE DATOS ");
   printf("\n ********************* \n");
   for (i=0; i<n ; i++) {
      printf("NOTAS ALUMNO %d - %s :\n", i, (alum+i)->nombre);
      for (j=0; j < (alum+i)->pnota->cantNotas ; j++) {
         printf("Nota %d: %d\n",  j,  *((alum+i)->pnota->notas+j) );
      }
   }

   return 0;
}
```

Argumentos o parámetros recibidos en el main por la línea de comando:

- Los programas que diseñamos en el curso generalmente en el `main` tienen argumento.

```
int main(int argc, char *argv[])
```

- La función `main` recibe como argumentos las opciones que se indican en la línea de comandos cuando ejecutas el programa desde la consola de GNU/Linux.
- `$ ./saluda -n nombre`

Argumentos o parámetros recibidos en el main por la línea de comando:

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
  if (argc != 3)
    printf ("Error: necesito que indiques el nombre con -n\n");
  else
    if (strcmp(argv [1], "-n") != 0)
      printf ("Error: sólo entiendo la opción -n\n");
    else
      printf ("Hola, %s.\n", argv [2]);
  return 0;
}
```